

DEVELOPMENT OF COMMUNICATION PROTOCOLS ON FPGA THROUGH PCIE

An Undergraduate Research Scholars Thesis

by

JOHN WRIGHT

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor:

Dr. Stavros Kalafatis

May 2020

Major: Electrical Engineering

TABLE OF CONTENTS

	Page
ABSTRACT.....	1
CHAPTER	
I. INTRODUCTION	2
Background	2
Overview	3
II. PROCESSES.....	4
III. ANTICIPATED OUTCOMES.....	10
REFERENCES.....	18

ABSTRACT

Development of Communication Protocols on FPGA through PCIe

John Wright
Department of Electrical and Computer Engineering
Texas A&M University

Research Advisor: Dr. Stavros Kalafatis
Department of Electrical and Computer Engineering
Texas A&M University

The research project I am proposing is an extension of a previous Texas A&M Senior Design Project completed in the spring of 2019 by four students. Their project was to develop protocols on FPGA through the use of PCIe. Their main areas of focus were on increasing bandwidth by aggregating PCIe lanes, reducing latency on performance critical accelerators and matching current PCIe transfer rate. My research will go even further with their project. I will take their previous design and streamline its features, improve performance mechanics and most importantly implement the package onto an FPGA. This purpose behind this project is to create a faster and higher performing PCIe protocol for data transfers, a key component in the large industry of data management and storage. The goals of this research project are to increase performance, decrease latency and improve the design of the previous project and implement it all on an FPGA board.

CHAPTER I

INTRODUCTION

The growing need for faster data processing and storage demands an upgrade to the current systems and technologies in place. One of these systems is PCIe, a data transfer system that has already begun to be improved upon with a senior design project completed in the spring of 2019. The improvements were made in the lane sizes and the protocol configurations but were never fully realized on an FPGA board. That is the focus of this research project in order to provide an overall improvement to satisfy the need for data speed.

Background

In today's world, there is an ever-increasing demand for faster data processing and improved memory access and storage. One of the main issues surrounding the desire to increase data transfer rates are the processor and accelerator configurations, which need to be optimized in order to reduce their current latency levels. The improvements to PCIe have been created by the previous team's project and will be improved and implemented on an FPGA in order to bring the project to completion. The need to improve the current PCIe protocols stems from the knowledge that CCIX protocols are much more effective in data transfers. The configuration of CCIX allows for shared memory spaces and differing data processors which stimulate better transfer rates with less latency. The goal of this research project is to improve and verify a protocol that would increase data transfer rates and decrease latency of PCIe communications, and to later implement the improved CCIX protocols as well.

Overview

This project will be a continuation of an updated PCIe system created to expand the bandwidth and reduce the latency of its data processing centers. Multiple subsystems were created by the previous team to achieve these goals. The system that has been created consists of introducing a new transaction layer to the physical and the data link layers of PCIe generation 4. The new layer starts with an Input Link Router module that takes in input from the four PCIe lanes and categorizes the input command as either a memory, message, configuration or I/O request. After this is completed, the ILR then converts the transaction layer package into a partial completion header and sends the package to the Hardware Subunit Input Buffer according to the type of command it is. The HSIB ensures each packet goes to the right pre-existing hardware subunit. The Output Link Router gets the packages from the hardware unit and sends them back to their respective PCIe lanes, and the Output Buffer stacks the packages on their way out.

This project will focus on the deliverable of the system implemented on an FPGA board for further verification. Additional areas of work will take place in improving the system of protocol inputs into the PCIe processors, as well as creating a self-generating protocol function for further automation within the verification of the overall product.

CHAPTER II

PROCESSES

This project is centered around the use of Quartus Prime software to program a Stratix V GX FPGA Development Kit in order to validate the overall system in hardware. The main version of software being used is Quartus Prime Standard Edition 18.1, which is the latest edition that still supports the Stratix V hardware. The first half of the research has focused on the correct application of the software to program the FPGA with the RTL code and also to configure the Hard IP block within the board in order for the PCIe edge connector to communicate with a host CPU. The remainder of the overall process is to communicate with the FPGA from a host CPU in order to send test packets of data to the RTL code on the board and measure the data speeds. Once the speeds and efficiencies have been documented, comparisons can be made to determine whether or not the new transaction layer performs better than existing PCIe specifications or not.

The standard speed for PCIe Gen 4 is 16 GT/s, and the goal is to obtain speeds up to 25 GT/s with the new protocol (“PCI Express® Base Specification”). In order to achieve this goal, the RTL code needs to be synthesized in Quartus and programmed onto the Stratix V FPGA. One modification that needed to be made to pass this stage was to rewrite the testbench file, since it failed to synthesize in the Quartus software. This was mainly due to the conflicts between blocking and non-blocking assignments within the code. Once the testbench was rewritten, it was validated by running it against the existing testbench to see if the overall functionality still remained the same.

```

C:\Users\John Wright\Documents\PCIe Research\FPGA Project>vvp ORIG.vvp
VCD info: dumpfile FPGA_Project_Test.vcd opened for output.
Lane 0 has enough output: Out =          4944, Needed =          4944
Lane 0 has passed =          4944
Lane 1 has enough output: Out =          4936, Needed =          4936
Lane 1 has passed =          4936
Lane 2 has enough output: Out =          5281, Needed =          5281
Lane 2 has passed =          5281
Lane 3 has enough output: Out =          4955, Needed =          4955
Lane 3 has passed =          4955

```

Figure A. Original Testbench Output

```

C:\Users\John Wright\Documents\PCIe Research\FPGA Project>vvp DEMO.vvp
VCD info: dumpfile Testbench_FPGA.vcd opened for output.
0 Waited
1 Waited
2 Waited
3 Waited
Lane 1 has passed      4936 of      4936 requests sent
Lane 0 has passed      4944 of      4944 requests sent
Lane 3 has passed      4955 of      4955 requests sent
Lane 2 has passed      5281 of      5281 requests sent

```

Figure B. Synthesized Testbench Output Matching Original

After validation was complete, the whole system was then programmed onto the FPGA. After multiple failed attempts to program the FPGA, the board was corrupted due to improper programming. The lack of documentation and information about how Quartus Prime operates with the Stratix V FPGA was the main setback in the programming stage. It took multiple weeks to get the correct process down as far as programming the second FPGA board correctly.

Once the RTL code was implemented on the second FPGA, the next stage was to configure the Hard IP block on the FPGA in order for a host computer to detect the FPGA's PCIe Edge Connector. This process meant researching how configuration works for an FPGA, and how device drivers are built in Linux since a driver would be needed to communicate with the FPGA from a CPU. After a few weeks work of finding multiple resources on Linux device

drivers, namely a Userspace I/O driver document, and different designs to configure the Hard IP block, the time came to program the FPGA board again with the configuration (Koch). Unfortunately, even though designs like the *Stratix V Avalon-MM Interface for PCIe Solutions User Guide* and the *Stratix V Avalon-ST Interface for PCIe Solutions User Guide* were specifically designed for the Stratix V GX FPGA board, both failed to compile in Quartus. One was due to the design being too big for the FPGA, and the other was due to outdated components.

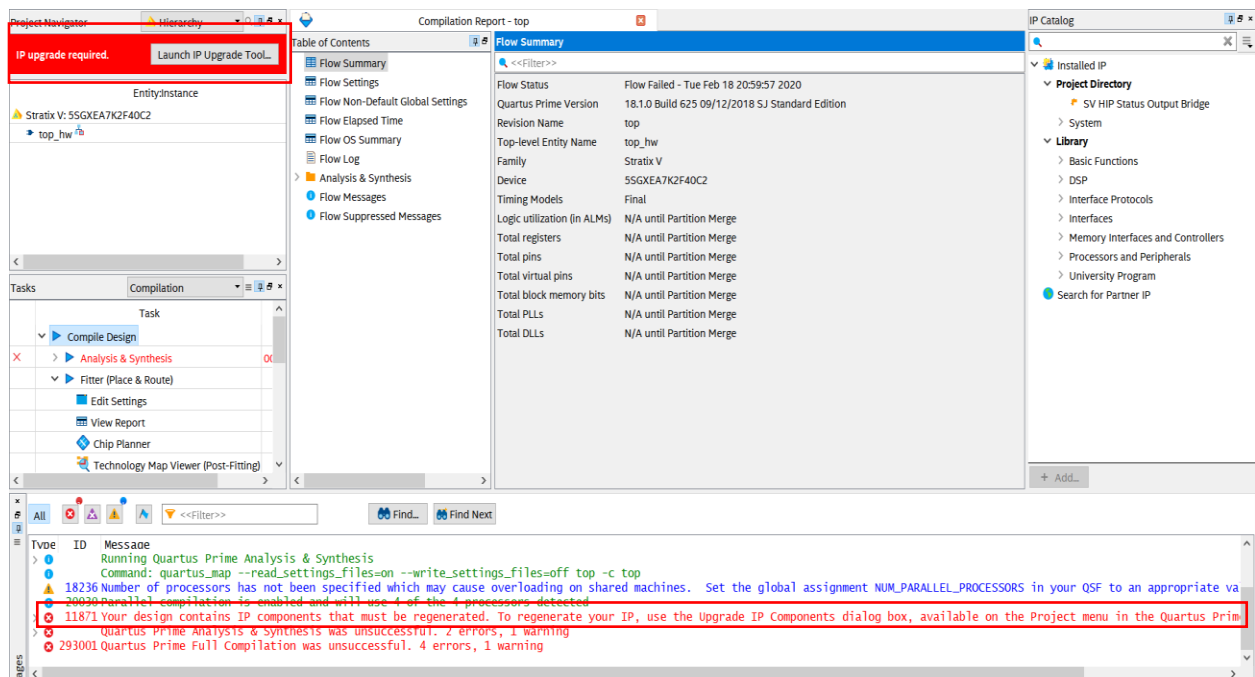


Figure C. Compilation Failure Due to Outdated Design

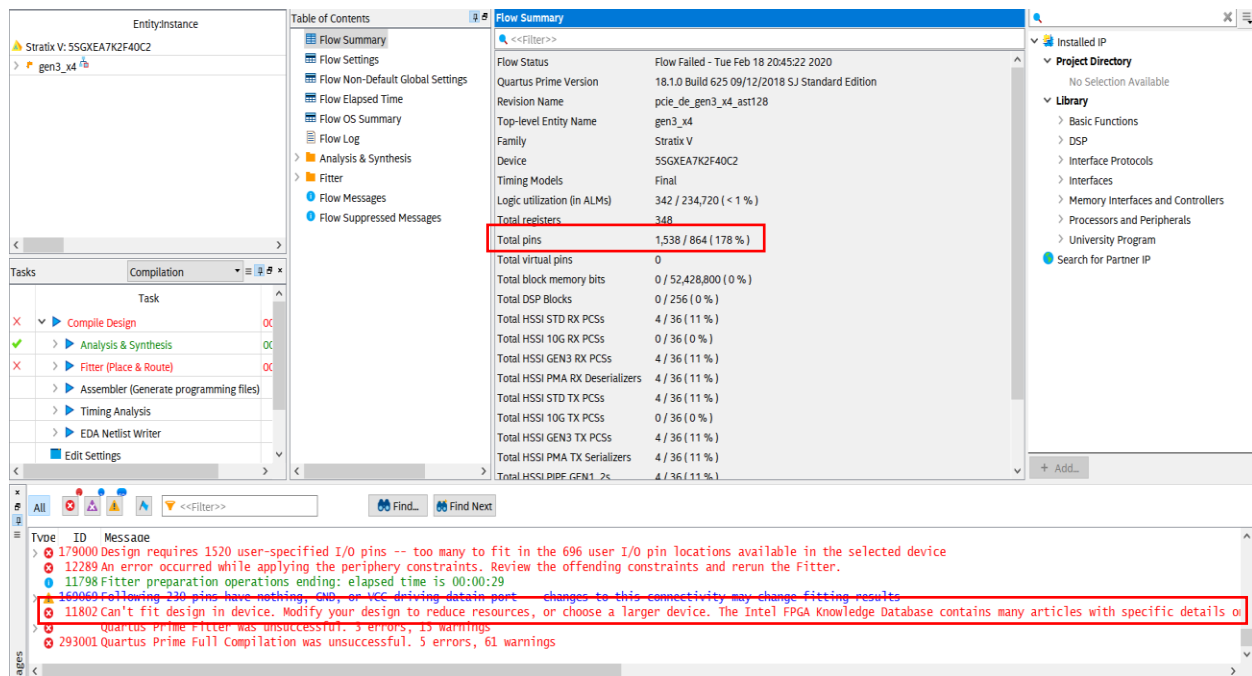


Figure D. Compilation Failure Due to Size of Design

After doing more research, a solution was found to use an older version of Quartus Prime (version 14.0) and this method did result in a successful compilation on the software. However, once the FPGA was connected for programming, any and all attempts to program the FPGA have failed, as Quartus is no longer able to detect the JTAG chain on the board. This resulted in another major setback, as solutions were being investigated. Many different methods to fix both FPGA board and get them to program again had failed, until the idea came from my TA Ping Wang to uninstall every version of Quartus except for the version most needed, in this case 14.0. After removing the other Quartus versions, the second FPGA board began to program successfully again. Multiple attempts to validate the fix proved that the solution was viable for the second FPGA board, but not for the first. There seems to be a different set of problems with the original FPGA board that still needs to be investigated further.

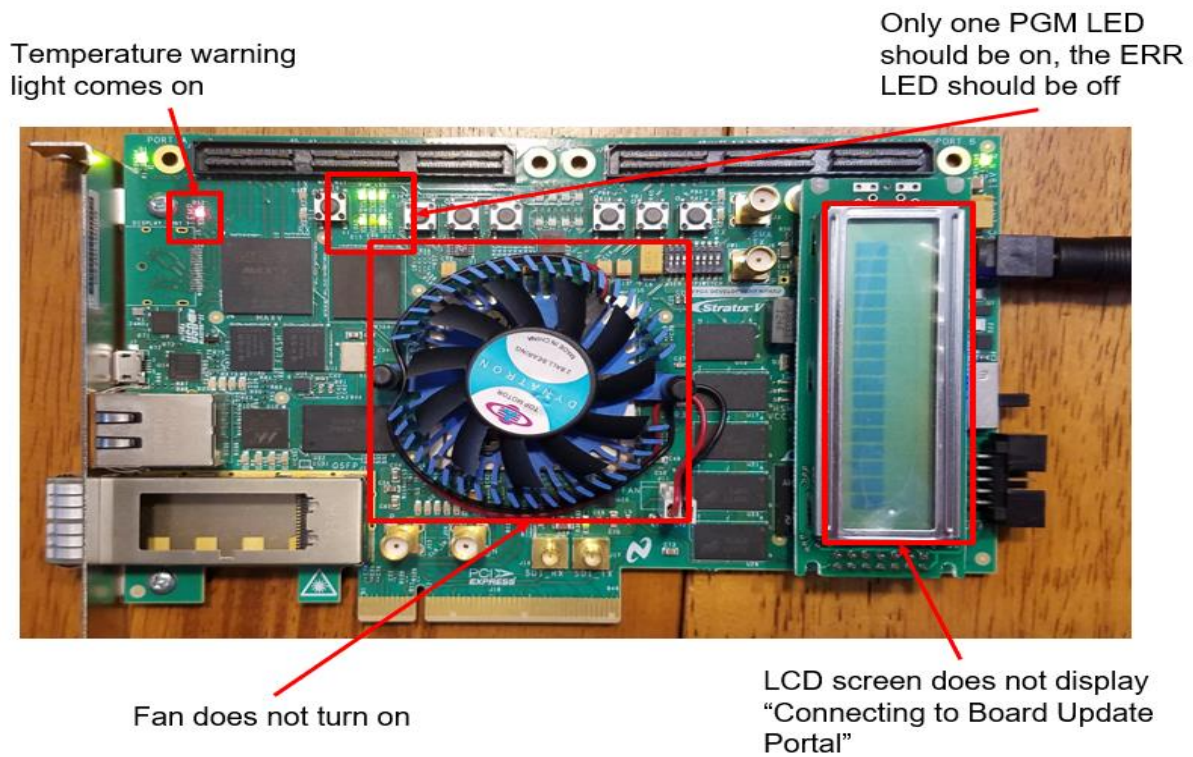


Figure E. First FPGA Board Corrupted State

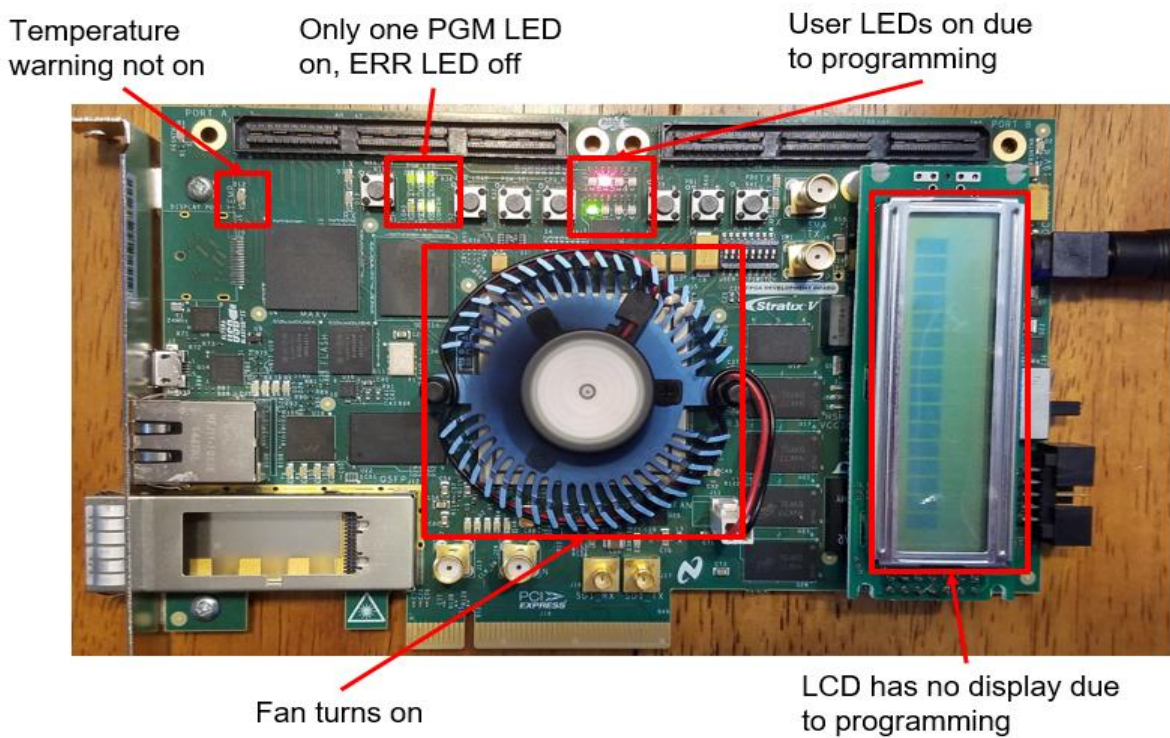


Figure F. Second FPGA Board Successful Program State

Due to the COVID-19 outbreak, all lab access has been shut down and complete data is unavailable at the time of publication for this URS thesis. The remaining steps needed to complete this research project include: connecting the RTL code to the pins for the PCIe edge connector, programing both the RTL code and the Hard IP configuration on the FPGA, using a Linux device driver with the testbench to run data packet simulations and collect data speed information, and comparing data speeds of RTL code to that of existing PCIe specifications. These items need to be performed with the help of a Linux CPU with an open PCIe slot which was in the lab that is now closed due to the virus. These final steps would allow for the final data collection necessary to draw conclusions on whether or not the new transaction layer is a better system than existing PCIe with a faster data transfer rate.

CHAPTER III

ANTICIPATED OUTCOMES

Final conclusions and results for this research project are incomplete at this time due to the virus outbreak causing the shutdown of all lab access for the remainder of the semester. Therefore, the overall outcome of the research is still up for speculation. The main objective was to create a more efficient transaction layer that would ultimately allow faster data transfer speeds than the current PCIe system. The various potential outcomes for this research would mean different things to the data industry as a whole.

On the upper end, speeds of 25 GT/s would be optimal, far surpassing PCIe Gen 4 speeds of 16 GT/s. However, this RTL code may not achieve the ideal results and could possibly only match existing PCIe speeds or even run at lower rates. Should the new RTL transaction layer only match existing speeds, there would be both positive outcomes and negative ones. It would mean a great deal to completely redesign the existing transaction layer and have it perform as well as PCIe Gen 4, even though the FPGA boards are only rated up to PCIe Gen 3. This would still be a success in terms of redesigning PCIe and showing how optimized clock cycles and distribution of data packets can improve upon a widely used system. With more research and testing it could provide an overall upgrade for future generations of data transfer systems, which would be a breakthrough in providing the growing demand for faster data with a viable solution. On the other hand, if this system only matches Gen 4 speeds or especially Gen 3 speeds, then it still needs a lot more work to go back and improve the RTL code again and again until it obtains more valuable results. This means more time spent on a system that may or may not prove to be

useful in the long run or may finally succeed when current technology has already surpassed its improvements.

Should the RTL transaction layer perform worse than existing PCIe specifications, then the value from this research to the industry would be to guide future improvement ideas and show how updated clock cycles with distributed data packets may not be the way to go. This project would serve to illustrate the unsuccessfulness that upgrading the transaction layer to optimize its components would have to be done differently if it is to work better than it currently does. However, should the RTL system outperform existing PCIe specs, mainly PCIe Gen 4, then this research proves to be much more beneficial to the industry by providing a working upgrade to the industry standard which companies could look at and try to expand upon. The increasing demand for faster data would meet with an increase in the speeds that the industry can produce to help stimulate growth in the big data world. Whatever the final outcome may be, this research project will provide an important piece of information to the tech industry. It will either serve as an example of how to optimize PCIe for the future or will serve as evidence that the redistribution of data packets and the algorithm that cuts down on unwanted clock cycles is not a path that leads to better overall performance, and that industry workers may need an alternative solution to their data transfer systems.

REFERENCES

Koch, Hans-Jürgen. “The Userspace I/O HOWTO.” The Userspace I/O HOWTO - The Linux Kernel Documentation, The Kernel Development Community, 11 Dec. 2006, www.kernel.org/doc/html/v4.13/driver-api/uio-howto.html.

“PCI Express High Performance Reference Design.” Intel Corporation, Dec. 2018.

“PCI Express* Avalon®-MM DMA Reference Design.” Intel Corporation, May 2017.

“PCI Express® Base Specification Revision 4.0 Version 0.3.” PCI-SIG, 19 Feb. 2014.

“Stratix V Avalon-MM Interface for PCIe Solutions User Guide.” Intel Corporation, 6 Nov. 2017.

“Stratix V Avalon-ST Interface for PCIe Solutions User Guide.” Intel Corporation, 20 Dec. 2019.

“Stratix V GX FPGA Development Board Reference Manual.” Altera Corporation, Sept. 2015.

“Stratix V GX FPGA Development Kit User Guide.” Altera Corporation, Dec. 2014.

“Stratix V Hard IP for PCI Express IP Core in the Quartus II 13.1 Software Release.” Altera Corporation, Dec. 2013.

“Stratix V Hard IP for PCI Express User Guide for the Avalon Memory-Mapped Interface with DMA.” Altera Corporation, Dec. 2013.

“Stratix V Hard IP for PCI Express User Guide.” Altera Corporation, June 2012.